

Large Language Models for Bio-image Analysis

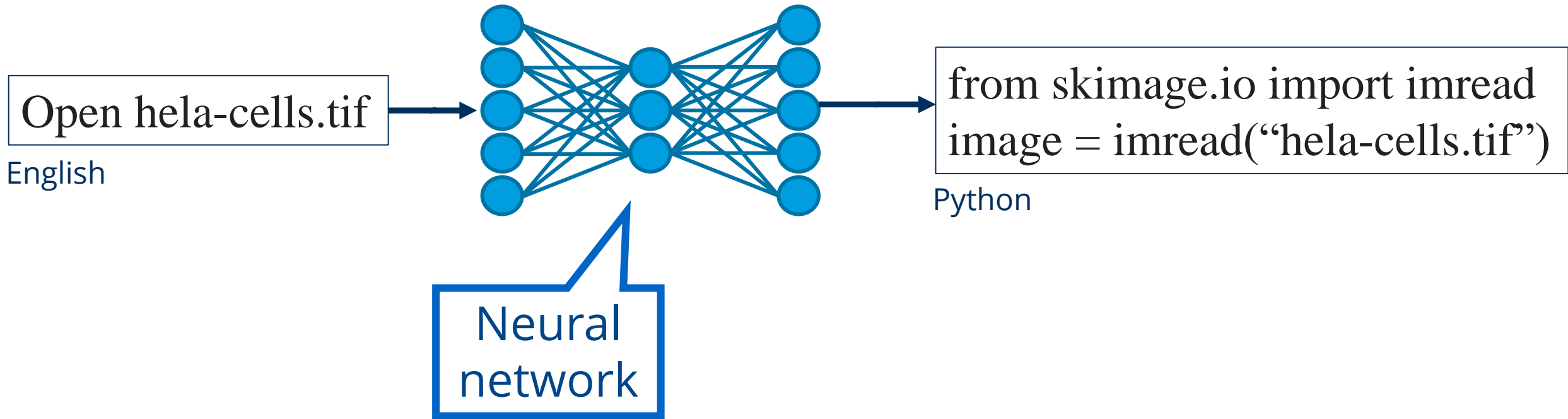
Robert Haase



<https://doi.org/10.5281/zenodo.11638980>

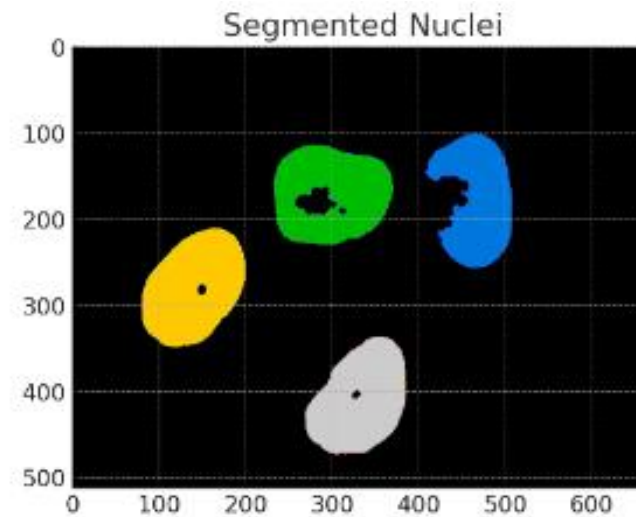
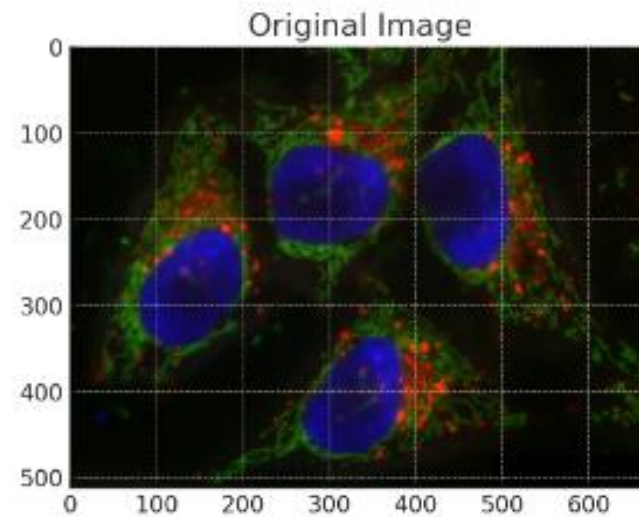
Large Language Models (LLMs)

Text-to-text, translation, code generation



Bio-image Analysis

My job ...

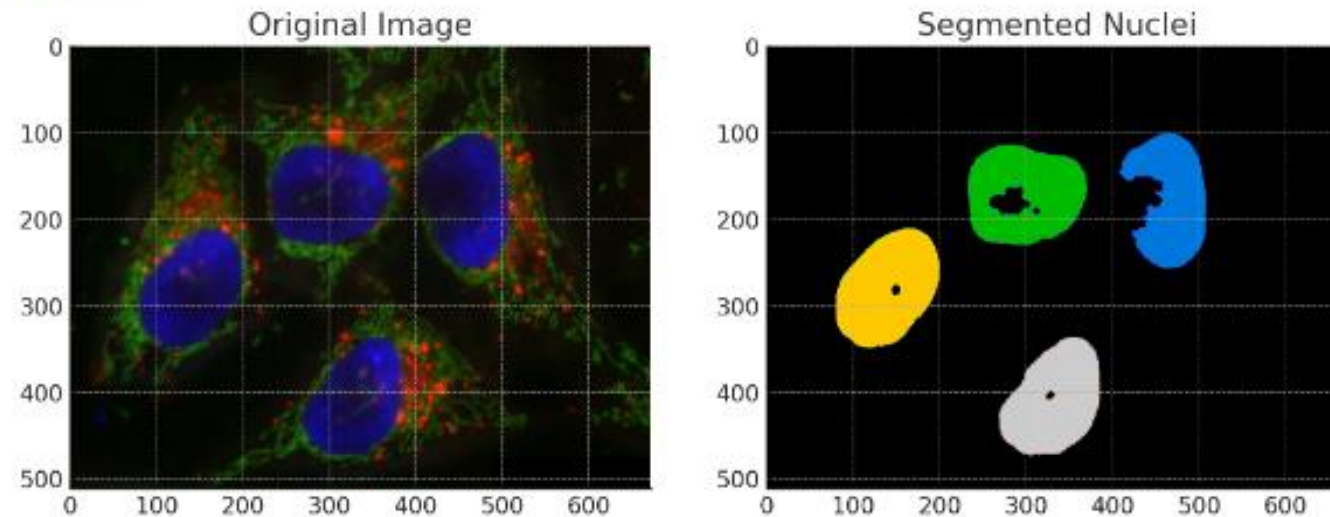


Bio-image Analysis using Large-Language Models

My job is changing, since we have ChatGPT

Prompt:
“Please segment
the nuclei in the
blue channel of
the image”

ChatGPT



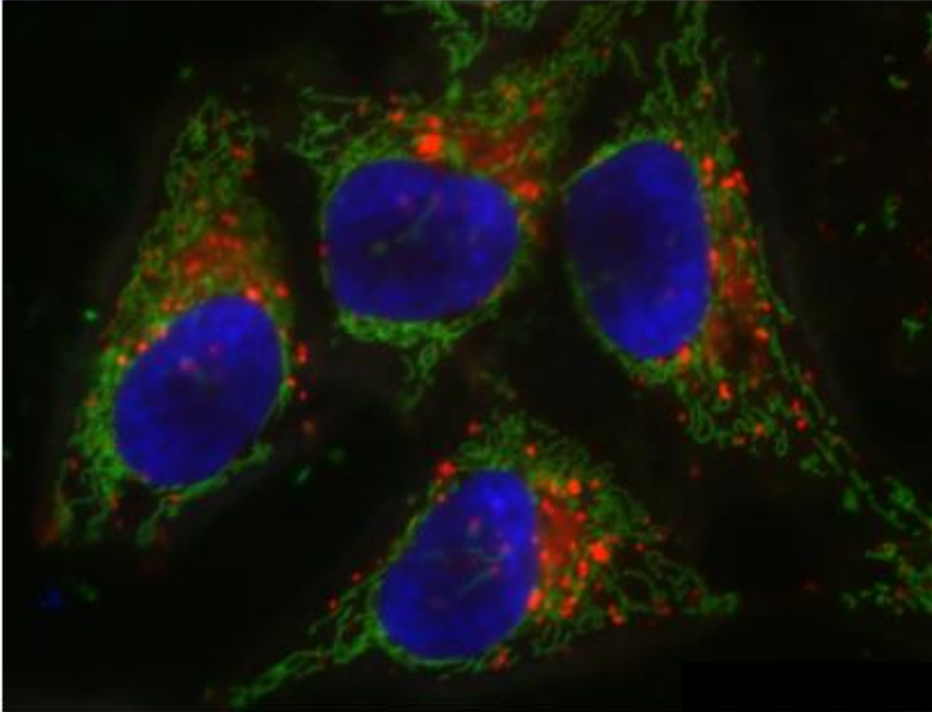
The segmentation of the nuclei has been performed on the blue channel of the image. The right side of the figure shows the segmented nuclei in different colors, each corresponding to a different nucleus in the field of view. Each color represents a unique label assigned to the individual nuclei detected in the image. [-]



Vision language models for counting objects

LLMs were not exactly built for this use-case...

You



How many blue nuclei are in this image?



ChatGPT

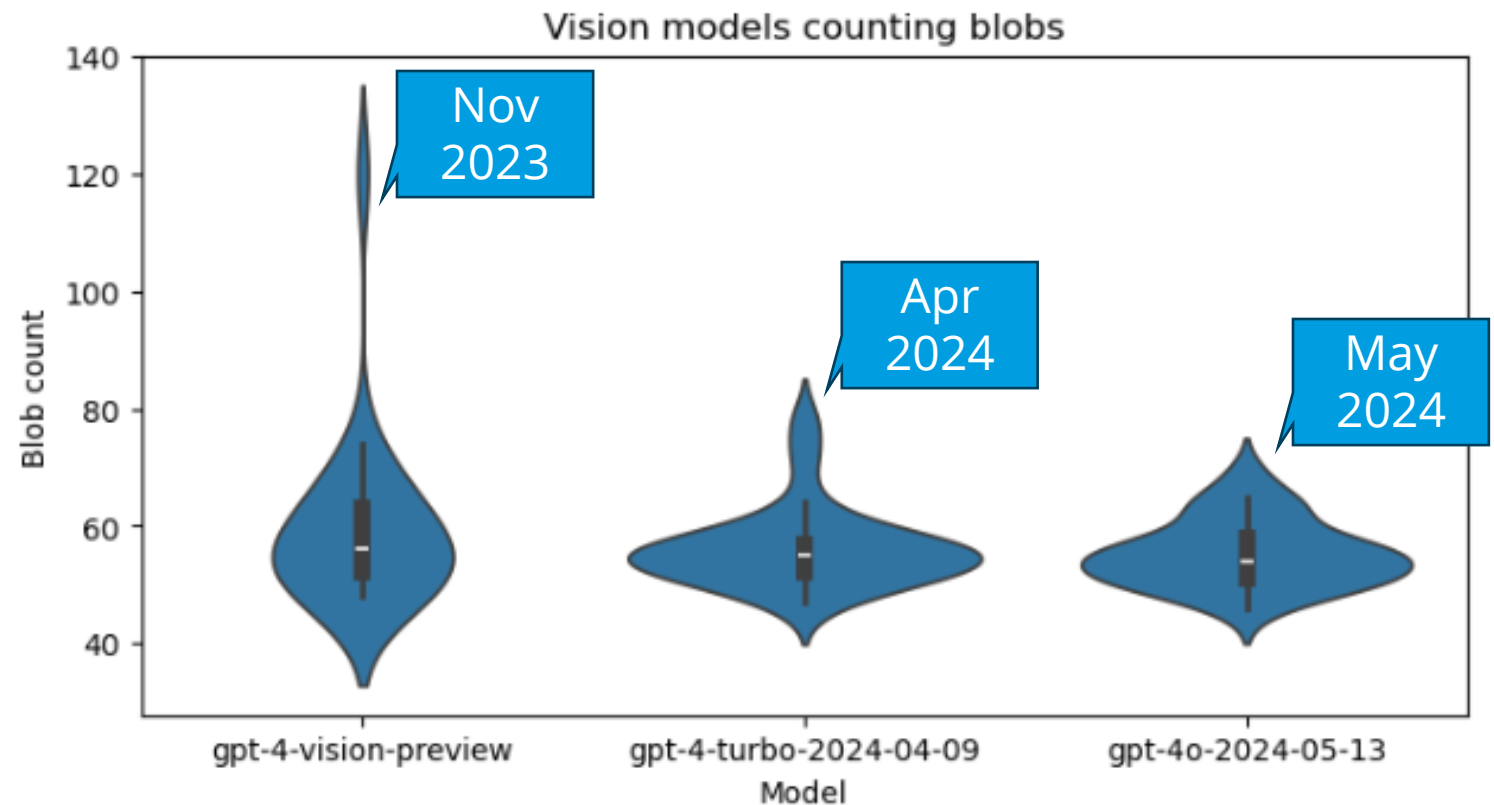
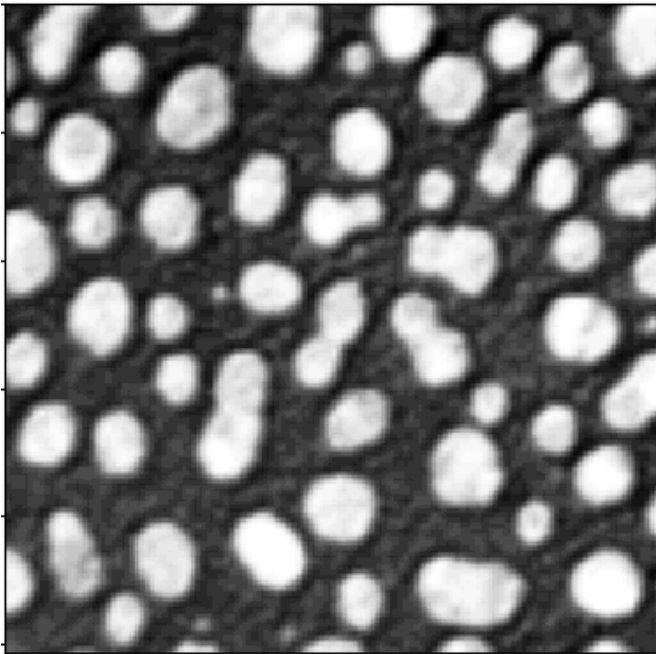
There are three blue nuclei visible in this image.



$n=1$

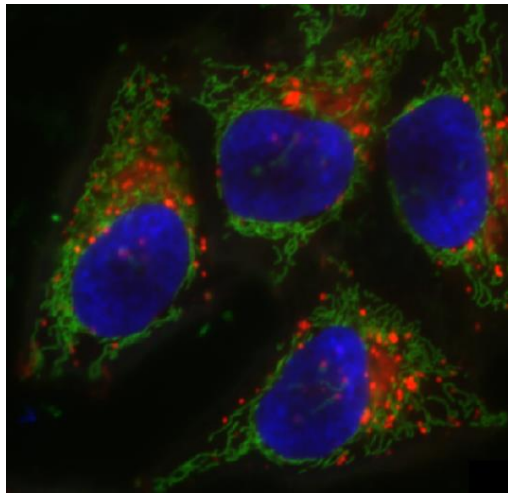
Vision language models for counting objects

Prompt: „Analyse the following image by counting the bright blobs. Respond with the number only. “ (n=25)

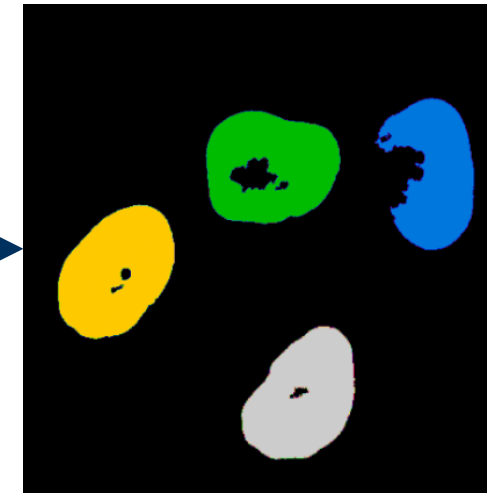
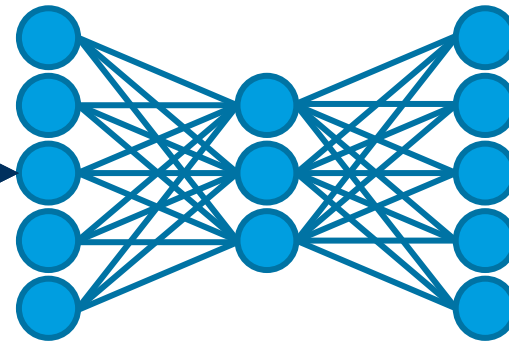


Multi-modal LLMs

Combining image, text and [...] data, to gain new [biological] insights.



How many cells are there?



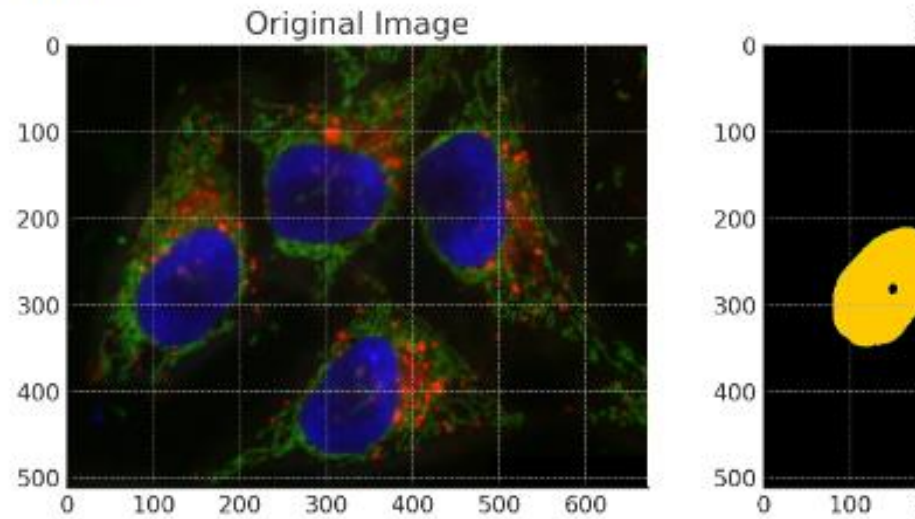
There are 4 cells.
I just marked their nuclei.

Bio-image Analysis using Large-L

My job is changing, since we have ChatGPT

Prompt:
“Please segment
the nuclei in the
blue channel of
the image”

ChatGPT



The segmentation of the nuclei has been performed on the blue channel
the figure shows the segmented nuclei in different colors, each corres
the field of view. Each color represents a unique label assigned to the
image. [~]



Analysis

```
from skimage import io, filters, measure, morphology
import matplotlib.pyplot as plt
import numpy as np

# Load the image
image = io.imread('/mnt/data/hela-cells-8bit.png')

# Since it's a color image, we'll just take the blue channel
blue_channel = image[:, :, 2]

# Threshold the image to get the binary image for nuclei
thresh = filters.threshold_otsu(blue_channel)
binary = blue_channel > thresh

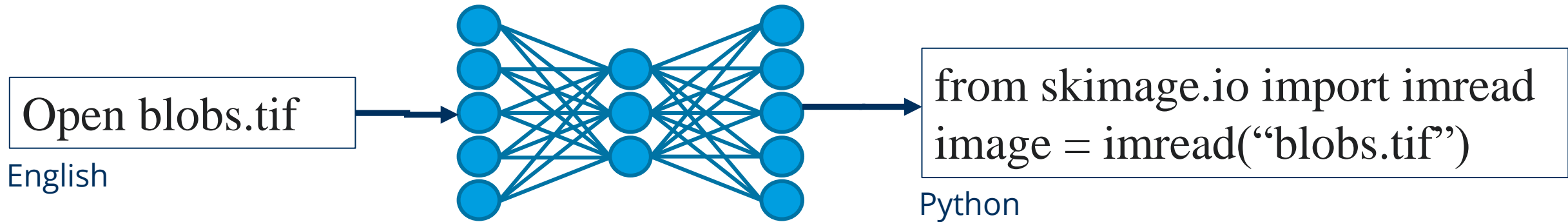
# Remove small objects and perform morphological closing to
cleaned = morphology.remove_small_objects(binary, 50)
closed = morphology.closing(cleaned, morphology.disk(3))

# Label the image
label_image = measure.label(closed)

# Plot the original image and the binary image
fig, ax = plt.subplots(1, 2, figsize=(12, 6))
ax[0].imshow(image)
ax[0].set_title('Original Image')
ax[1].imshow(label_image, cmap='nipy_spectral')
ax[1].set_title('Segmented Nuclei')
plt.show()
```


Large Language Models (LLMs)

Text-to-text, translation, code generation



Open source assistant: BiA-Bob



- No need to upload your [image] data (-> privacy)
- Multi-agent / -platform capable
- Vision models
- Extensible through plugins
- Integrated in Jupyter



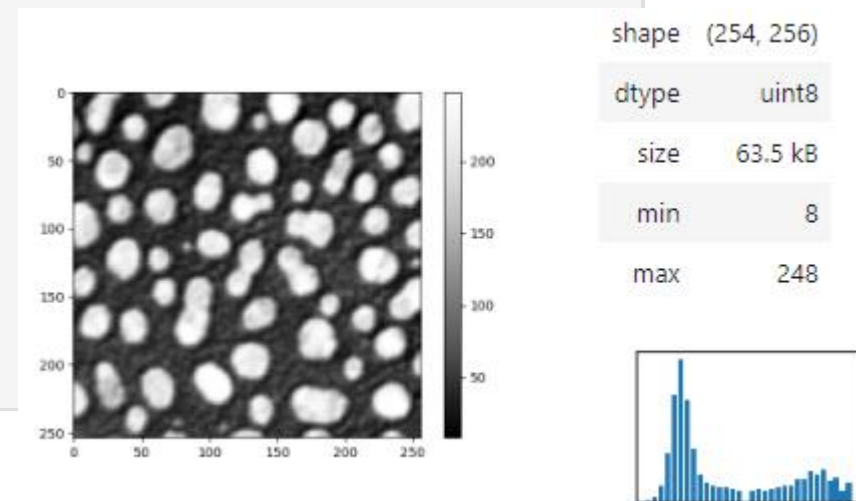
```
%bob Load the blobs.tif image file and show it.
```

I will load the `blobs.tif` image file and then display it using the appropriate visualization function.

```
from skimage.io import imread
import stackview
```

```
# Load the image file
filename = 'blobs.tif'
image = imread(filename)
```

```
# Display the image
stackview.insight(image)
```



Generating complex code blocks



Use the `%%bob ...` syntax to ask for complex analysis tasks.

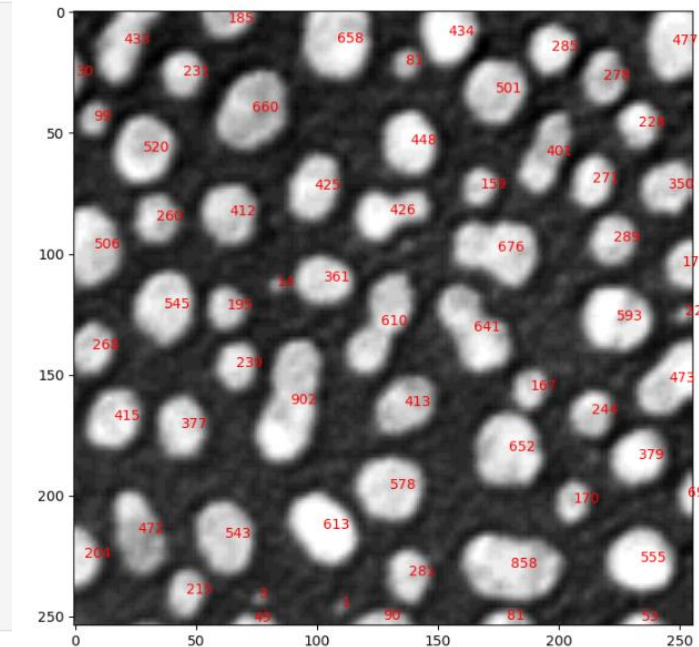
In [4]:

```
%%bob
```

```
Load the blobs.tif grayscale image from disc.  
Segment the nuclei-like objects in it using  
Otsu's method and connected component labeling.  
Measure area of the segmented labels.  
Print the area on top of the labels.
```

In [5]:

```
from skimage.io import imread  
from skimage.filters import threshold_otsu  
from skimage.measure import label, regionprops  
import matplotlib.pyplot as plt  
import stackview  
  
# Load the image  
image = imread('blobs.tif')  
  
# Apply Otsu's thresholding  
threshold = threshold_otsu(image)  
binary_image = image > threshold  
  
# Perform connected component labeling  
labels = label(binary_image)  
  
# Measure area of the segmented labels  
props = regionprops(labels)  
areas = [prop.area for prop in props]  
  
# Create an image with labels and area values on top  
fig, ax = plt.subplots(figsize=(8, 8))  
ax.imshow(image, cmap='gray')  
  
for prop in props:  
    y, x = prop.centroid  
    area = prop.area  
    ax.text(x, y, f'{area}', color='red')  
  
stackview.curtain(image, labels)
```



Generating notebooks

... also great for learning Python



```
%%bob assume you communicate with a Python beginner. Generate a Jupyter notebook named `python_weather_analysis` that covers
* Load german_weather_2023.csv using pandas,
* visualize the head of the table,
* summarize the table and show the infos for the dataframe
* compute the mean and maximum temperature (a column in the table)
* make use of pandas internal plotting methods to plot the rain over the days (scatter plot), omit the x-axis labels.
* group the data to the four seasons by associating the months of a year.
* plot a boxplot of rain in the four seasons using seaborn.'
```

A notebook has been saved as `python_weather_analysis.ipynb`.

Generating notebooks

... also great for learning Python

In these kind of tasks, Python and ChatGPT are !

Python Weather Analysis

In this notebook, we will perform a basic weather data analysis using Python. We will:

1. Load weather data from a CSV file using pandas.
2. Visualize the first few rows of the table.
3. Summarize the table and show the information of the dataframe.
4. Compute the mean and maximum temperature from the data.
5. Create a scatter plot of rain over the days using pandas plotting methods.
6. Group the data by seasons and plot a boxplot of the rain data for the four seasons using seaborn.

Disclaimer

This code is generated by an AI model using the [bia-bob](#) project. It is good scientific practice to check the code and results carefully.

Import Libraries

First, we will import the necessary libraries for our analysis.

```
In [1]: import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Load Weather Data

We will load the weather data from a CSV file called `german_weather_2023.csv` using pandas.

```
In [2]: df = pd.read_csv('german_weather_2023.csv')
```

Visualize the Head of the Table

Let's have a look at the first few rows of the dataframe to understand the structure of the data.

```
In [3]: display(df.head())
```

| | date | temperature | rain |
|---|------------|-------------|-----------|
| 0 | 2023-01-01 | -1.254599 | 14.507143 |
| 1 | 2023-01-02 | 0.986585 | 6.560186 |
| 2 | 2023-01-03 | -4.419164 | 13.661761 |
| 3 | 2023-01-04 | 2.080726 | 5.205845 |
| 4 | 2023-01-05 | 3.324426 | 7.123391 |

Summarize the Table and Show Info

We will summarize the dataframe and show its info to understand the columns and types of data we are dealing with.

```
In [4]: print(df.describe())
```

| | temperature | rain |
|-------|-------------|------------|
| count | 365.000000 | 365.000000 |
| mean | 10.708104 | 10.959650 |
| std | 8.545935 | 7.639665 |
| min | -4.944779 | 0.145447 |
| 25% | 5.143935 | 6.272566 |
| 50% | 10.427244 | 9.456826 |
| 75% | 15.182317 | 13.961197 |
| max | 29.949553 | 57.799883 |

```
In [5]: print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 365 entries, 0 to 364
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  --
 0   date    365 non-null      obj
 1   temperature  365 non-null     float64
 2   rain    365 non-null     float64
dtypes: float64(2), object(1)
memory usage: 8.7+ KB
None
```

Compute Mean and Max

Next, we will compute the mean and max

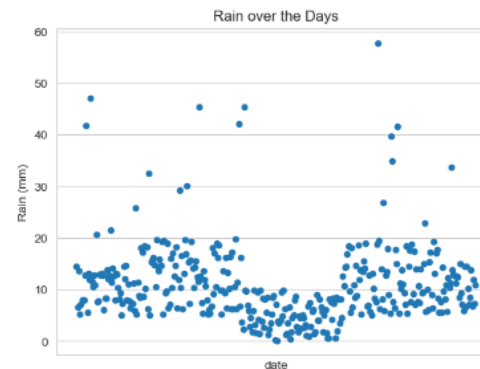
```
In [6]: mean_temperature = df['temperature'].mean()
max_temperature = df['temperature'].max()
print(f'Mean Temperature: {mean_temperature}')
print(f'Maximum Temperature: {max_temperature}')
```

```
Mean Temperature: 10.70810411661468
Maximum Temperature: 29.94955255618853
```

Scatter Plot of Rain Over Days

We will use pandas' internal plotting methods to create a scatter plot of rain over the days, omitting the

```
In [9]: df.plot.scatter(x='date', y='rain', xlabel='date', ylabel='Rain (mm)', title='Rain over Days')
plt.xticks([])
plt.show()
```

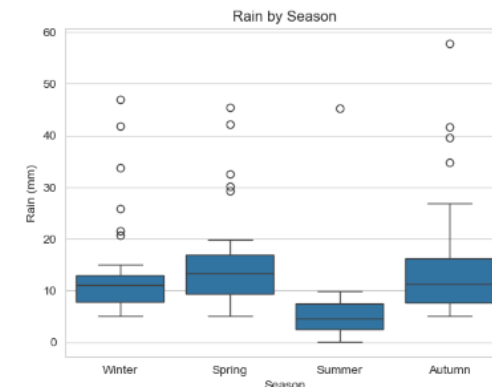


Group Data by Seasons

We will group the data by seasons (Winter, Spring, Summer, Autumn) by associating the months of the year and then plot a boxplot of rain in the four seasons using seaborn.

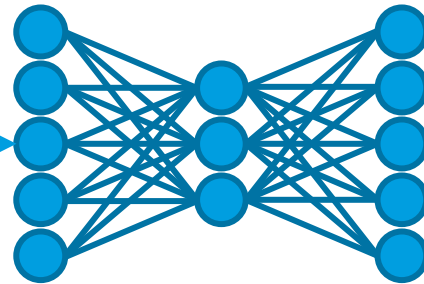
```
In [8]: def get_season(month):
    if month in [12, 1, 2]:
        return 'Winter'
    elif month in [3, 4, 5]:
        return 'Spring'
    elif month in [6, 7, 8]:
        return 'Summer'
    else:
        return 'Autumn'

df['season'] = pd.to_datetime(df['date']).dt.month.apply(get_season)
sns.boxplot(x='season', y='rain', data=df)
plt.title('Rain by Season')
plt.xlabel('Season')
plt.ylabel('Rain (mm)')
plt.show()
```



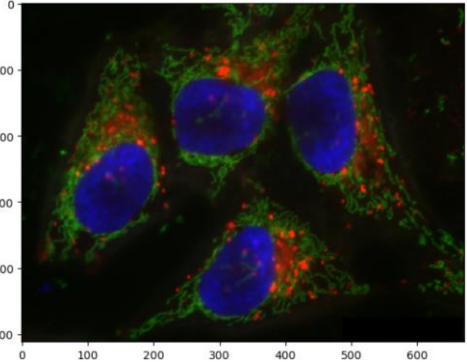
Vision Language Models

- Classifying images 🤔
- Describing images



A picture of a
cat and a
microscope

Vision language models



```
[3]: %%bob image  
what's in this microscopy image? Answer in one short sentence.
```

The microscopy image shows cells with blue-stained nuclei and green-stained cytoskeletal structures.

```
[5]: %%bob What deep-learning based algorithm could segment the nuclei ?
```

The deep-learning-based algorithm that could segment the nuclei is Cellpose.

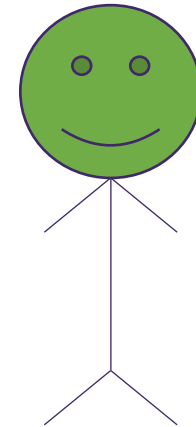
```
[6]: %%bob Write code for applying Cellpose to this image.
```

We will apply the Cellpose algorithm to segment nuclei in the image using Python code.

```
# Apply Cellpose model  
model = models.Cellpose(gpu=False, model_type='nuclei')  
masks, flows, styles, diams = model.eval(image, diameter=None, channels
```

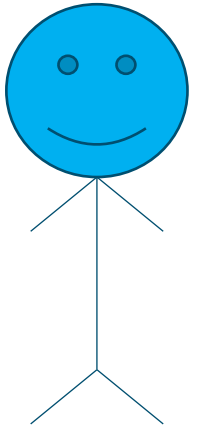


That's half right!



Biologist

That makes sense!



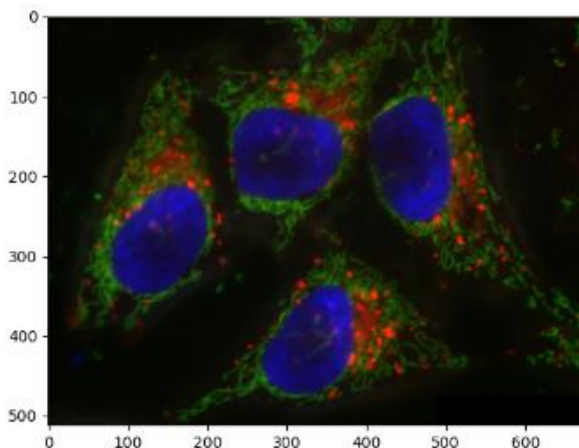
Computer scientist

Generating notebooks using vision models



Ask %%bob to generate a Jupyter notebook

```
hela_cells = imread("hela-cells-8bit.tif")
stackview.insight(hela_cells)
```



shape (512, 672, 3)
dtype uint8
size 1008.0 kB
min 0
max 255



Present Bob an image like this

```
%%bob hela_cells
Please write a new Jupyter notebook for processing this image.
Its filename is `hela-cells-8bit.tif`.
At the beginning of the notebook describe the image it is made for.
I would like to segment the objects in the blue channel.
Write Python-code for doing this and please add explanatory notebook
cells in between explaining what you're doing in detail as I'm a
Python-beginner.

Thanks :-)
```

A notebook has been saved as `segmenting_hela_cells_blue_channel.ipynb`.

Generating notebooks using vision models



The screenshot displays two Jupyter Notebook windows. The left window, titled 'generate_notebooks.ipynb', contains a section 'Generating a notebook using vision capabilities'. It explains that a model can generate a notebook from a prompt. The prompt asks the model to create a notebook for segmenting HeLa cells in the blue channel of an image named 'hela-cells-8bit.tif'. The model's response is shown in a code cell, which includes the following Python code:

```
[3]: hela_cells = imread("hela-cells-8bit.tif")
stackview.insight(hela_cells)
```

The output of this code is a visualization of the HeLa cells image, showing the blue channel segmented. The image is displayed with its shape (512, 672, 3), dtype (uint8), size (1008.0 kB), and a histogram of the pixel values.

The right window, titled 'segmenting_hela_cells_blue_channel.ipynb', contains a section 'HeLa Cells Image Processing'. It explains that the notebook is designed to process an image of HeLa cells. The image is a composite containing different color channels: blue, green, and red. The objective is to segment the objects in the blue channel, which typically represents the nuclei of the cells. The notebook will use the `skimage` (scikit-image) library and `matplotlib` for image processing and plotting.

The notebook then shows the code for importing necessary libraries and loading the image:

```
[1]: import numpy as np
import matplotlib.pyplot as plt
from skimage import io, filters, measure, morphology, color

# Set a figure size
plt.rcParams['figure.figsize'] = [10, 8]

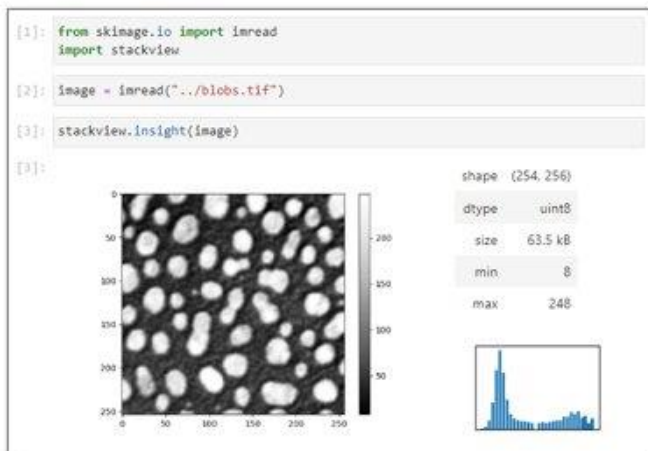
# Read the image
image = io.imread('hela-cells-8bit.tif')

# Display the original image
plt.imshow(image)
plt.title('Original Image')
plt.axis('off')
plt.show()
```

The output of this code is a visualization of the original HeLa cells image, showing the blue channel segmented.

Optimizing documentation

E.g. to make code more readable



```
from bia_bob import bob
```

`%bob` modify the notebook `mod_notebook.ipynb` by adding explanatory markdown cells between the code cells. Also add a markdown cell at the very beginning that introduces the entire notebook shortly.

The modified notebook has been saved as `mod_notebook_1.ipynb`.

Introduction

In this notebook, we will demonstrate how to load a bioimage, visualize it using stackview, and explore the image interactively. We will be using an image of blobs for this example. The steps include loading the image and visualizing it using different stackview functionalities.

Step 1: Import necessary libraries

We will start by importing the necessary libraries. In this case, `skimage.io` for loading the image and `stackview` for visualization.

```
from skimage.io import imread
import stackview
```

Step 2: Load the image

Next, we will load the image file into a variable. The image used in this notebook is `blobs.tif`.

```
image = imread("../blobs.tif")
```

Step 3: Visualize the image using stackview insight

We will use `stackview.insight` to visualize the loaded image. This function allows for inspecting the image.

```
stackview.insight(image)
```



GPU-Accelerating code



Original:

```
[7]: %%acc
from skimage.io import imread
from skimage.filters import threshold_otsu
from skimage.measure import label
from skimage.color import label2rgb
import stackview

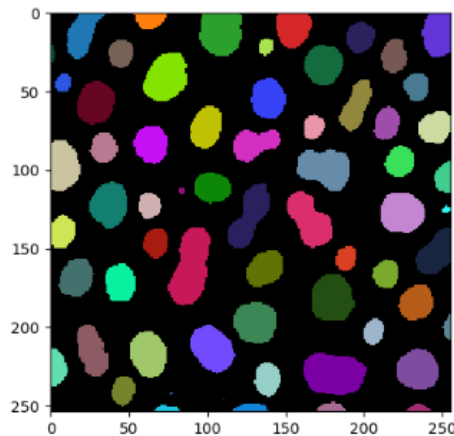
# Load the image
image = imread('blobs.tif')

# Apply global Otsu threshold to segment the image
binary_image = image > threshold_otsu(image)

# Label the objects
labels = label(binary_image)

# Display the color labels
stackview.insight(labels)
```

[7]:



| | |
|-------|------------|
| shape | (254, 256) |
| dtype | int32 |
| size | 254.0 kB |
| min | 0 |
| max | 64 |

Replacement:

```
[8]: import pyclesperanto_prototype as cle
import stackview

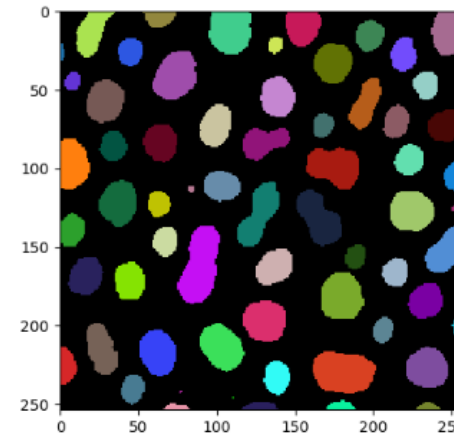
# Load the image
image = cle.imread('blobs.tif')

# Apply global Otsu threshold to segment the image
binary_image = cle.threshold_otsu(image)

# Label the objects
labels = cle.connected_components_labeling_box(binary_image)

# Display the color labels
stackview.insight(labels)
```

[8]:



It's our
responsibility
that results are
comparable!

https://link.springer.com/chapter/10.1007/978-3-030-76394-7_5

Under the hood: prompt engineering



Context-dependent
system prompt
considering

- local variables and functions
- installed python libraries
- chat history

+ your prompt

You are a extremely talented bioimage analyst and you use Python to solve your tasks ...

...

Python specific code snippets
If the user asks for those simple tasks, use these code snippets.

* Load an image file from disc and store it in a variable:
```\n`

```
from skimage.io import imread
image = imread(filename)
```\n`
```

...

Todos

Answer your response in three sections:

1. Summary: First provide a short summary of the task.
2. Plan: Provide a concise step-by-step plan without any code.
3. Code: Provide the code.

About
6500
tokens
(≈words)

Compatible language models



- ChatGPT by OpenAI

```
[9]: bob.initialize("gpt-4-1106-preview")

[10]: %%bob
load blobs.tif,
segment it using Voronoi-Otsu-Labeling in pyclesperanto_prototype
and show the image and the resulting label image together in one matplotlib subplot
```

I will load the `blobs.tif` image, segment it using the Voronoi-Otsu-Labeling method from the `pyclesperanto_prototype` library, and display the original image and the resulting labeled image together in a single matplotlib subplot.

```
[11]: from skimage.io import imread
import pyclesperanto_prototype as cle
import matplotlib.pyplot as plt

# Load the image
image = imread('blobs.tif')

# Segment the image with Voronoi-Otsu-Labeling
labels = cle.voronoi_otsu_labeling(image, spot_sigma=2, outline_sigma=2)
```

- Gemini by Google

```
[2]: bob.initialize("gemini-pro")

[3]: %%bob
load blobs.tif,
segment it using Voronoi-Otsu-Labeling in pyclesperanto_prototype
and show the image and the resulting label image together in one matplotlib subplot
```

Given the image "blobs.tif," follow these steps to load it, segment it using Voronoi-Otsu-Labeling, and visualize the original image and the resulting label image in a single matplotlib subplot.

```
[4]: import matplotlib.pyplot as plt
from pyclesperanto_prototype import voronoi_otsu_labeling

# Load the image
In = plt.imread('blobs.tif')

# Segment the image using Voronoi-Otsu-Labeling
Out = voronoi_otsu_labeling(In)
```

Compatible language models



- Mistral through Blablador

```
[5]: bob.initialize(  
      endpoint='blablador',  
      model='Mistral-7B-Instruct-v0.2')
```

```
[6]: %bob load blobs.tif and show it
```

There was an issue. Retrying (1/3)...
Load and display the "blobs.tif" image file.

```
[7]: import io  
      import matplotlib.pyplot as plt  
      import skimage.io as io  
  
      image = io.imread('blobs.tif')  
      plt.imshow(image)  
      plt.show()
```

Free online
service for
German
academics,
offered by
HelmholtzAI

Open Source:
[https://github.com/
HelmholtzAI-
FZJ/FastChat](https://github.com/HelmholtzAI-FZJ/FastChat)

Many OpenSource models *seem* less
capable than the commercial models,
when it comes to complex tasks

- CodeLLama via ollama

```
[2]: bob.initialize(endpoint='ollama', model='codellama')
```

Afterwards, you can use `bob` as usual.

```
[3]: %bob load blobs.tif and show it
```

The task is to load a TIFF image file named "blobs" and display its
content on screen.

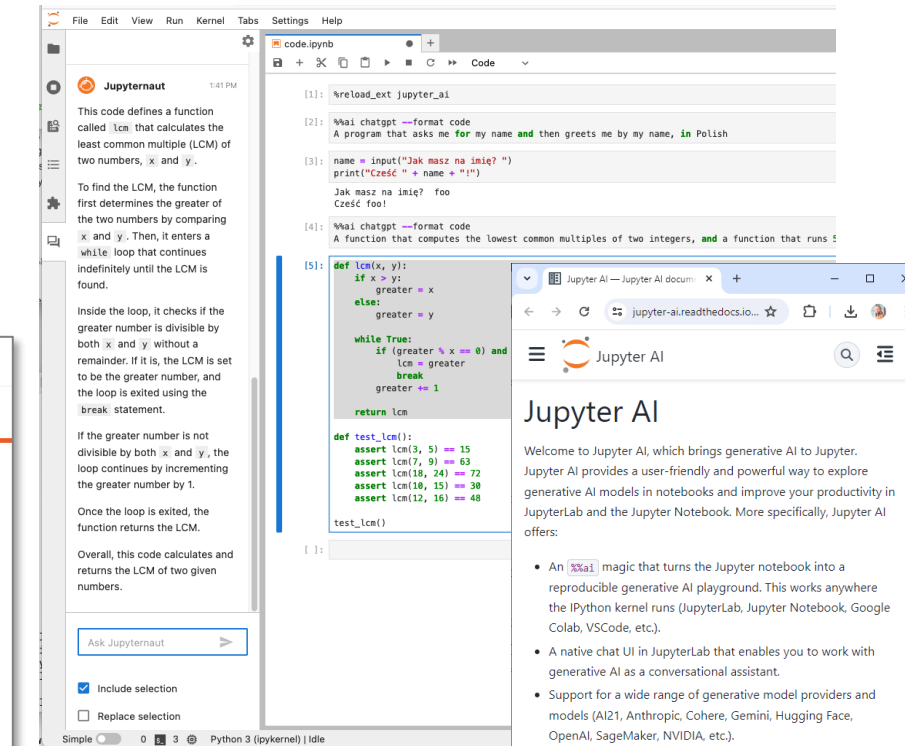
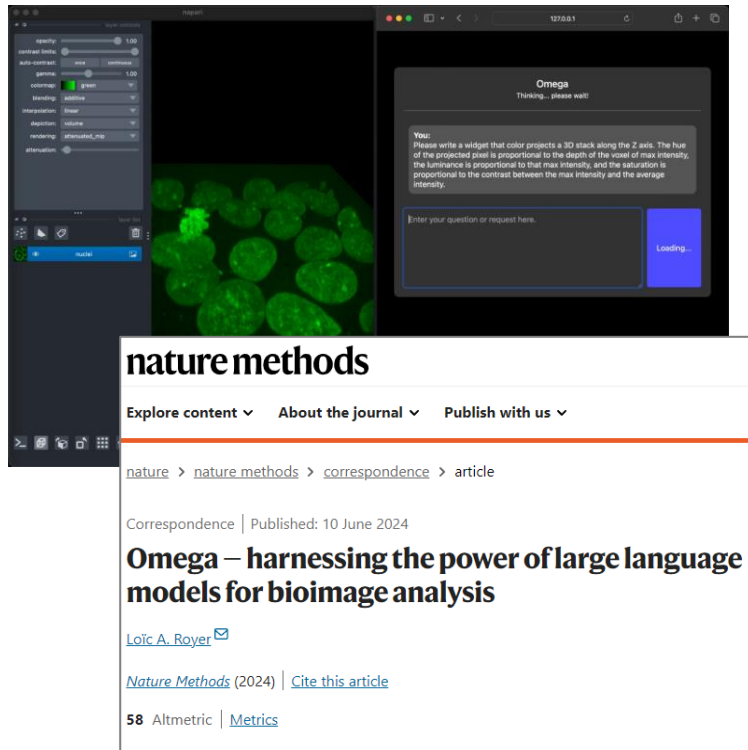
```
[4]: import numpy as np  
      from skimage import io  
      from matplotlib import pyplot as plt  
  
      # Load the TIFF image  
      img = io.imread('blobs.tif')  
  
      # Display the image using imshow() from matplotlib  
      plt.imshow(img)  
      plt.show()
```

Runs on
your local
computer
(offline)

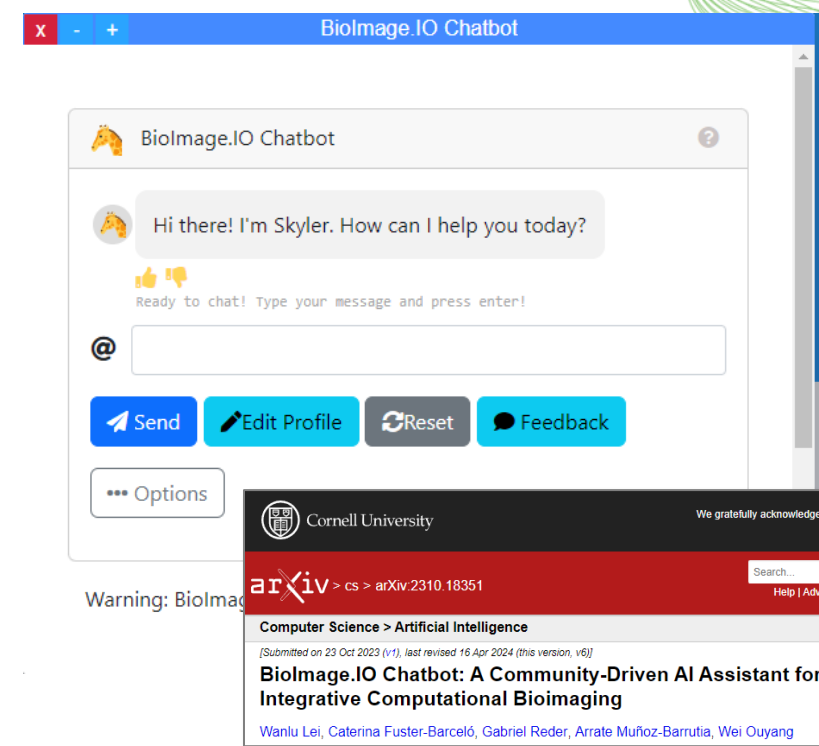
Similar tools

Bob is not alone... and they are becoming more.

Napari-chatGPT / Omega



Bioimage-io ChatBot




<https://github.com/royerlab/napari-chatgpt>

<https://jupyter-ai.readthedocs.io/>

<https://bioimage.io/#/>
<https://arxiv.org/abs/2310.18351>

Similar tools

Bob is not alone... and they are becoming more.



Microscope Image Analysis GPT

By Antonios Lioutas

Expert in microscopy image analysis with Python, skilled in tools like scikit-image, SimpleITK, Cellpose, Napari, Starfish, Dask, Numpy and Pandas.

★ 4.2
Ratings (10+)

Other
Category

300+
Conversations

Conversation Starters

How do I use Cellpose for cell segmentation?

What's the best way to analyze fluorescence images in Python?

Can you help me debug this SimpleITK code?

Tips for improving image analysis with Dask?

Capabilities

- ✓ Code Interpreter & Data Analysis
- ✓ Browsing

<https://chatgpt.com/g/g-FGdNx7Mll-microscope-image-analysis-gpt>

GPTs


Discover and create custom versions of ChatGPT that combine instructions, extra knowledge, and any combination of skills.

image analysis

All

- Image to Video**
Image analysis and video creation
By community builder · 10K+
- Smarter Image Analysis**
Deep and powerful image analysis, returning more accurate and relevant details th...
By Finn J · 1K+
- Bio-image Analysis GPT**
Bio-image Analysis with Python, a GPT created with content from the BioImageAna...
By Robert Haase · 1K+
- PPT and PDF Analyst with Image Analysis**
Professional tone, analyzing PPTs, PDFs, and imagery.
By fuying ding · 1K+
- Microscope Image Analysis GPT**
Expert in microscopy image analysis with Python, skilled in tools like scikit-image, S...
By Antonios Lioutas · 300+
- Image Analyse and Description for Recreation**
Provides detailed forensic and artistic image analysis and will try to recreate the im...
By Andreas Dötsch · 500+
- Stable Diffuser**
Expert image analysis, image-to-image, and image & prompt generation for DALL...
By Kenneth Whelan · 900+
- SR image analysis**
I am analysing the spatial distribution of siglec and IgM molecules . I have images ...
By Vivek Ravikumar · 20+
- Image Analysis Assistant**
图像分析助理，专注于细节和深度分析
By fisherdaddy.com · 7

<http://chat.openai.com/gpts>



Bio-image Analysis GPT

By Robert Haase

Bio-image Analysis with Python, a GPT created with content from the BioImageAnalysisNotebooks by R. Haase, G. Witz, M. Fernandes, M.L. Zoccoler, S. Taylor, M. Lampert, T. Korten, licensed CC-BY 4.0 and BSD3 unless mentioned otherwise.
<https://haesleinhuepf.github.io/BioImageAnalysisNotebooks>

★ 4.2
Ratings (20+)

Other
Category

1K+
Conversations

Conversation Starters

How can I segment bright blobs in a fluorescence microscopy...

How can I segment cells in a fluorescence microscopy image...

How can I measure the number of neighbors in a label image?

How can I erode labels in a label image?

Capabilities

- ✓ Browsing

<https://chatgpt.com/g/g-psAohb1OY-bio-image-analysis-gpt>

Benchmarking LLMs for Bio-image Analysis

Which LLM should I pay for?

Is GPT-4o better than Claude?

Do LLMs introduce bias?

Can LLMs act according to good scientific practice?

How can we validate LLMs for BiA?

How *good* are LLMs for Bio-image Analysis?

Will we all be replaced by AI anytime soon?

What are the limits of the LLM technology?

How can we validate specific LLM output without ground truth?

Benchmarking LLMs for Bio-image Analysis

Example test-case inspired by HumaEval (Chen et al 2021)

```
[1]: def workflow_segmentation_measurement_summary(image):  
    """  
    This function implements a workflow consisting of these steps:  
    * threshold intensity input image using Otsu's method  
    * label connected components  
    * measure area of the labeled objects  
    * determine mean area of all objects  
    """  
  
    import skimage  
    import numpy as np  
    binary_image = image > skimage.filters.threshold_otsu(image)  
    label_image = skimage.measure.label(binary_image)  
    stats = skimage.measure.regionprops(label_image)  
    areas = [s.area for s in stats]  
    return np.mean(areas)
```

Prompt

Reference
solution

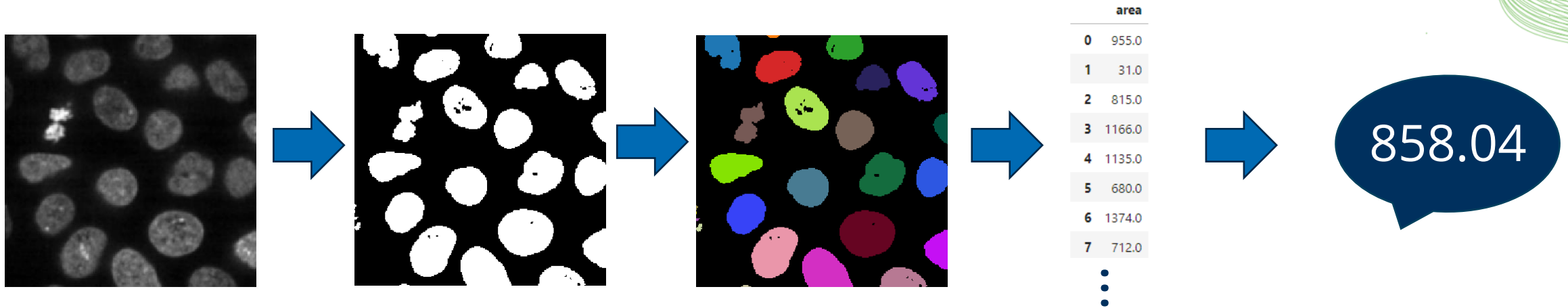
```
[2]: def check(candidate):  
    import numpy as np  
  
    assert candidate(np.asarray([  
        [0,0,0,0,0],  
        [1,1,1,0,0],  
        [1,1,1,0,0],  
        [1,1,0,0,0],  
        [0,0,0,0,0],  
    ])) == 8
```

Unit test
(excerpt)

We formulated
57 of such test-
cases (yet)

Benchmarking LLMs for Bio-image Analysis

Use case: segment the image and measure the average area of objects.



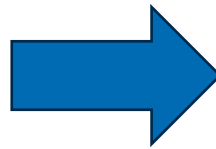
Unit-test pass-rate (n=10):

| | reference | gpt-4-turbo-2024-04-09 | Claude-3-opus-20240229 | gpt-4-1106-preview | gpt-3.5-turbo-1106 | gemini-pro | codellama |
|---|-----------|------------------------|------------------------|--------------------|--------------------|------------|-----------|
| workflow_segmentation_measurement_summary | 1.0 | 0.9 | 1.0 | 0.8 | 0.5 | 0.5 | 0.1 |

Benchmarking LLMs for Bio-image Analysis

Use-case: compute the correlation matrix

| | a | b | c | d | e |
|----|----------|----------|----------|----------|----------|
| 0 | 1.600000 | 0.100000 | 1.600000 | 1.700000 | 1.700000 |
| 1 | 2.300000 | 0.200000 | 2.300000 | 2.400000 | 2.400000 |
| 2 | 2.600000 | 0.300000 | 2.600000 | 2.400000 | 2.400000 |
| 3 | 3.700000 | 0.300000 | 3.700000 | 3.600000 | 3.600000 |
| 4 | 3.400000 | 0.400000 | 3.400000 | 3.500000 | 3.500000 |
| 5 | 3.900000 | 0.400000 | 3.900000 | 3.900000 | 3.900000 |
| 6 | 4.300000 | 0.400000 | 4.300000 | 4.400000 | 4.400000 |
| 7 | 4.300000 | 0.500000 | 4.300000 | 4.200000 | 4.200000 |
| 8 | 4.000000 | 0.500000 | 4.000000 | 4.100000 | 4.100000 |
| 9 | 5.100000 | 0.500000 | 5.100000 | 5.000000 | 5.000000 |
| 10 | 5.200000 | 0.600000 | 5.200000 | 5.100000 | 5.100000 |
| 11 | 5.300000 | 0.600000 | 5.300000 | 5.400000 | 5.400000 |
| 12 | 5.500000 | 0.600000 | 5.400000 | 5.600000 | 5.600000 |



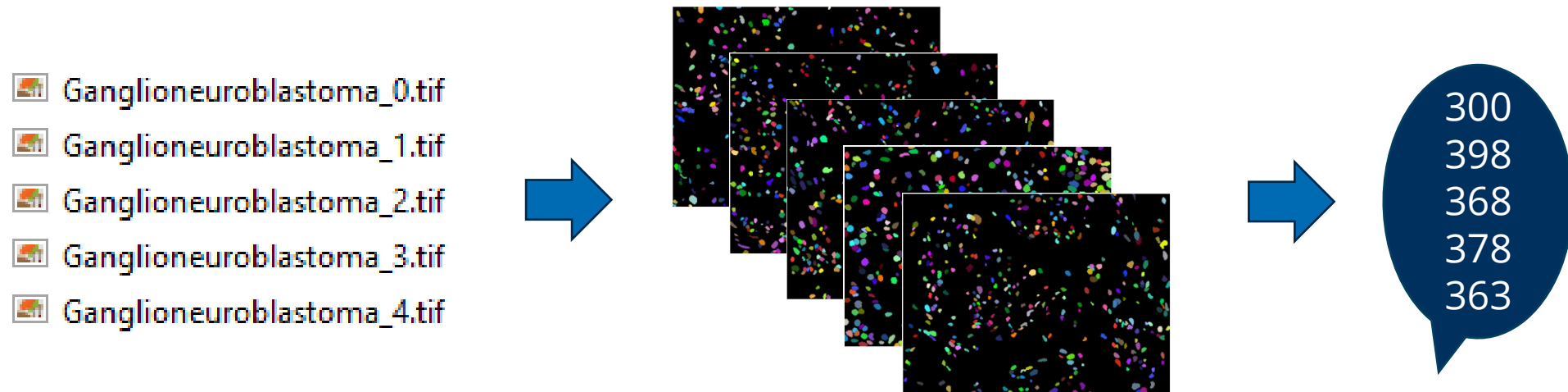
| | a | b | c | d | e |
|---|----------|----------|----------|----------|----------|
| a | 1.000000 | 0.949504 | 0.999775 | 0.995800 | 0.995800 |
| b | 0.949504 | 1.000000 | 0.949594 | 0.946039 | 0.946039 |
| c | 0.999775 | 0.949594 | 1.000000 | 0.995001 | 0.995001 |
| d | 0.995800 | 0.946039 | 0.995001 | 1.000000 | 1.000000 |
| e | 0.995800 | 0.946039 | 0.995001 | 1.000000 | 1.000000 |

Unit-test pass-rate (n=10):

| | reference | gpt-4-turbo-2024-04-09 | Claude-3-opus-20240229 | gpt-4-1106-preview | gpt-3.5-turbo-1106 | gemini-pro | codellama |
|------------------------------|-----------|------------------------|------------------------|--------------------|--------------------|------------|-----------|
| pair_wise_correlation_matrix | 1.0 | 1.0 | 1.0 | 0.9 | 1.0 | 0.5 | 0.1 |

Benchmarking LLMs for Bio-image Analysis

Use case: Count segmented objects in a folder of segmentation results.



Unit-test pass-rate (n=10):

| | reference | gpt-4-turbo-2024-04-09 | Claude-3-opus-20240229 | gpt-4-1106-preview | gpt-3.5-turbo-1106 | gemini-pro | codellama |
|--|-----------|------------------------|------------------------|--------------------|--------------------|------------|-----------|
| workflow_batch_process_folder_count_labels | 1.0 | 0.1 | 0.0 | 0.3 | 0.0 | 0.0 | 0.0 |

Benchmarking LLMs for Bio-image Analysis

Unit-test pass-rate (n=10)

| | reference | gpt-4-turbo- 2024-04-09 | Claude-3-opus- 20240229 | gpt-4-1106- preview | gpt-3.5-turbo- 1106 | gemini-pro | codellama |
|--|-----------|----------------------------|----------------------------|------------------------|------------------------|------------|-----------|
|--|-----------|----------------------------|----------------------------|------------------------|------------------------|------------|-----------|

Statistics / tabular data wrangling

| | | | | | | | |
|---------------------------|-----|-----|-----|-----|-----|-----|-----|
| combine_columns_of_tables | 1.0 | 0.8 | 0.1 | 1.0 | 0.9 | 0.7 | 0.1 |
| create_umap | 1.0 | 0.8 | 1.0 | 0.9 | 1.0 | 0.8 | 0.0 |
| t_test | 1.0 | 1.0 | 1.0 | 0.9 | 1.0 | 0.5 | 0.3 |

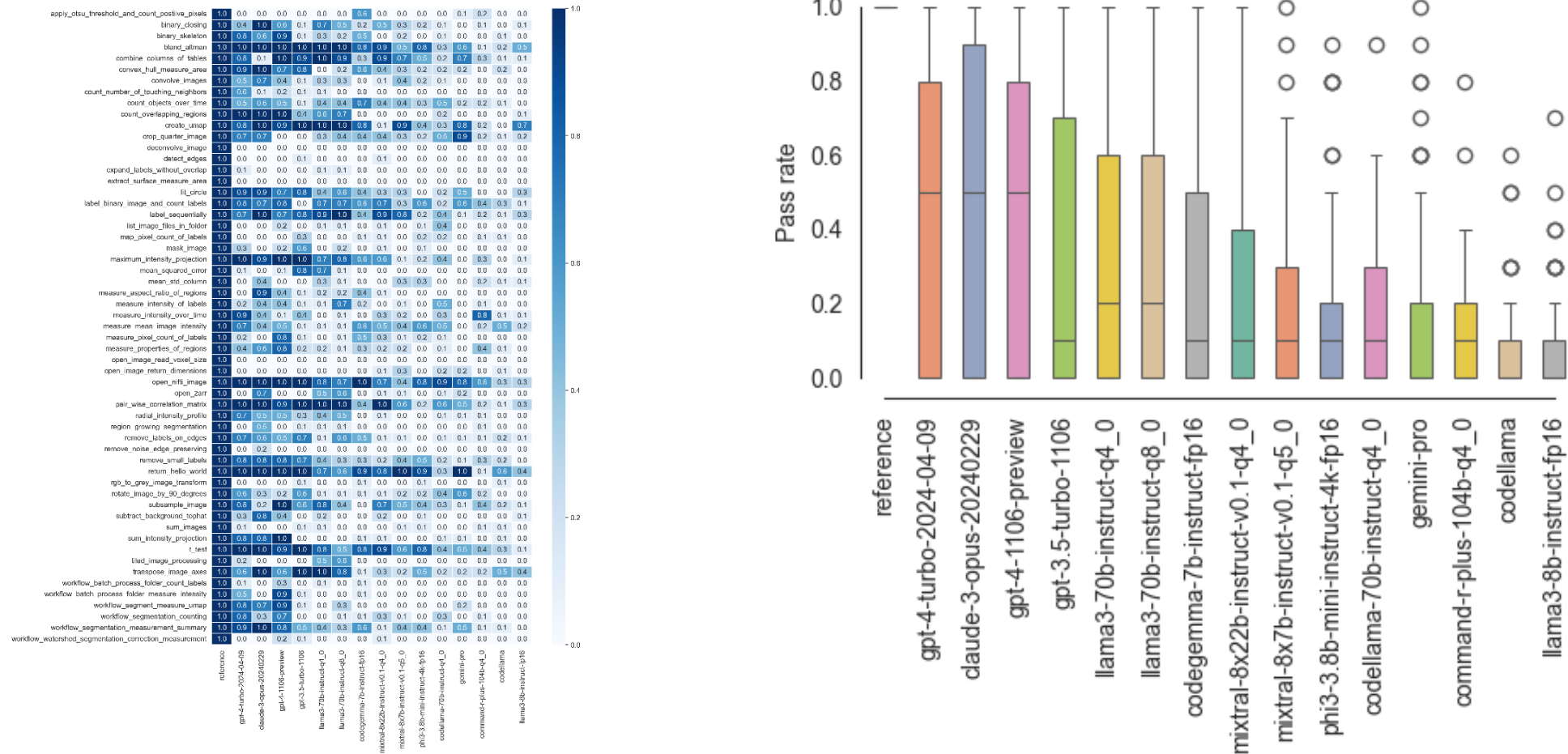
Measurements / feature extraction

| | | | | | | | |
|------------------------------------|-----|-----|-----|-----|-----|-----|-----|
| measure_intensity_over_time | 1.0 | 0.9 | 0.4 | 0.1 | 0.4 | 0.0 | 0.1 |
| measure_intensity_of_labels | 1.0 | 0.2 | 0.4 | 0.4 | 0.1 | 0.0 | 0.0 |
| measure_properties_of_regions | 1.0 | 0.4 | 0.6 | 0.8 | 0.2 | 0.0 | 0.1 |
| count_number_of_touching_neighbors | 1.0 | 0.6 | 0.1 | 0.2 | 0.1 | 0.0 | 0.0 |

Advanced workflows / big data

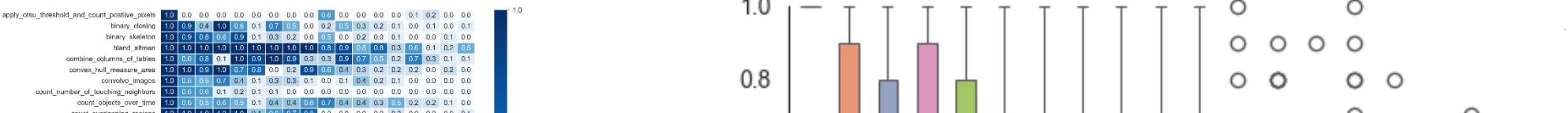
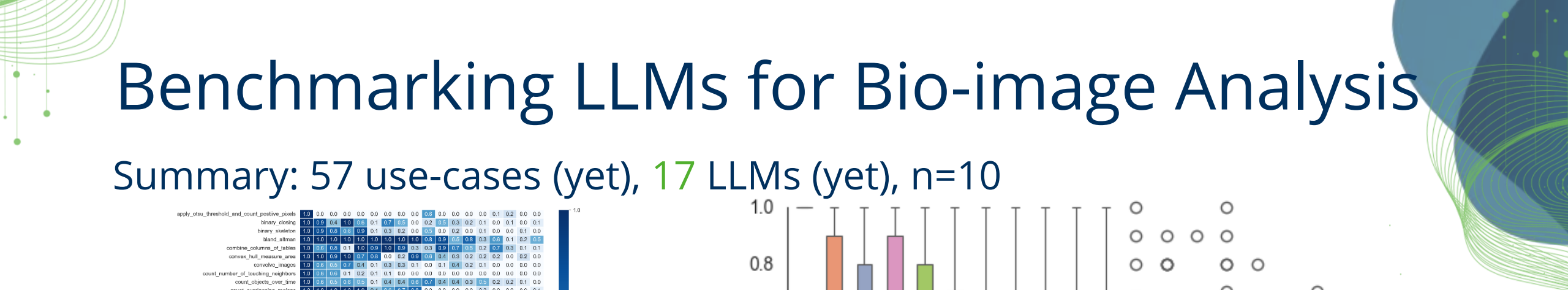
| | | | | | | | |
|---|-----|-----|-----|-----|-----|-----|-----|
| tiled_image_processing | 1.0 | 0.2 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| workflow_batch_process_folder_measure_intensity | 1.0 | 0.5 | 0.0 | 0.9 | 0.1 | 0.0 | 0.0 |

Summary: 57 use-cases (yet), 15 LLMs (yet), n=10



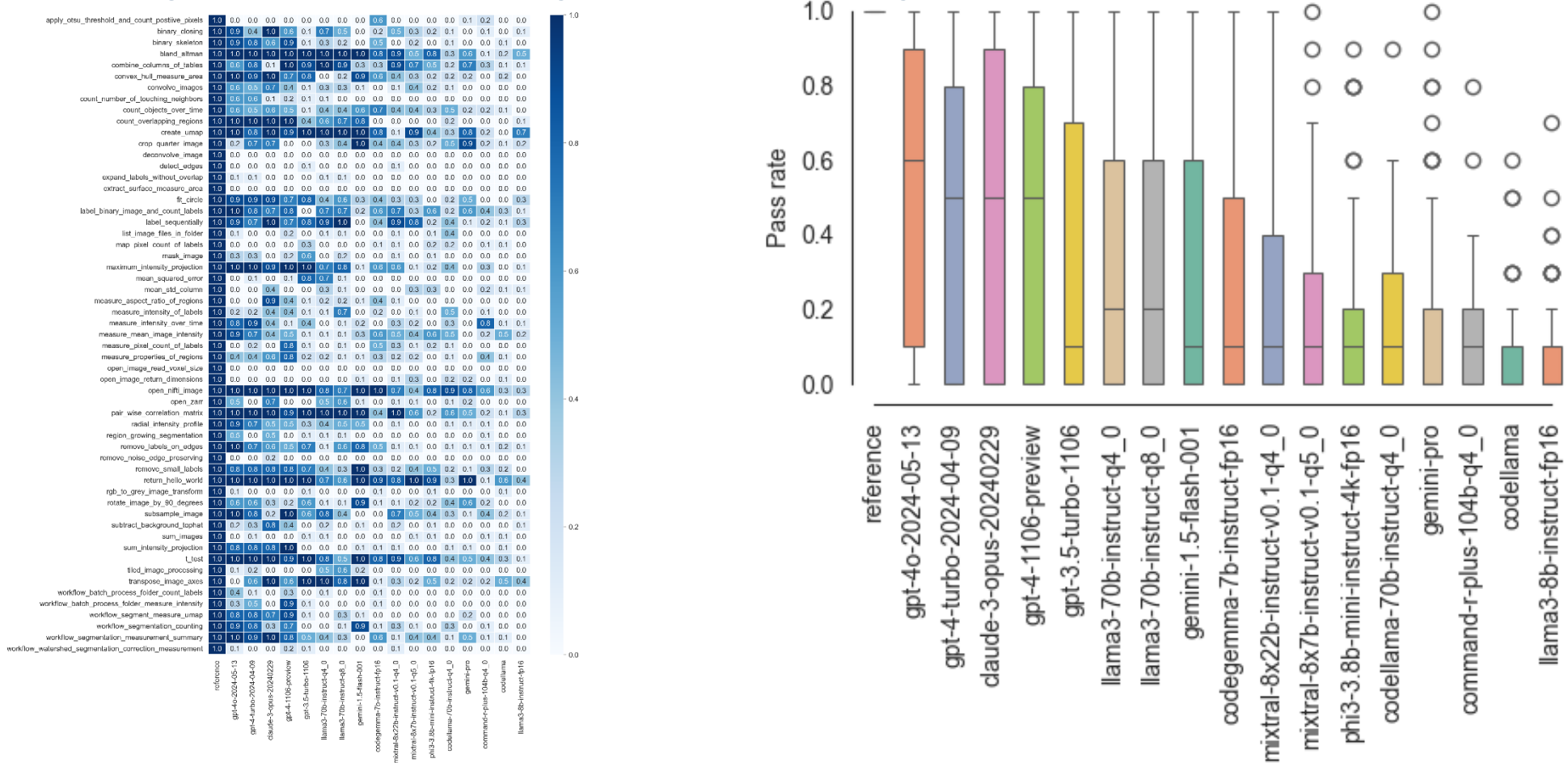
Benchmarking LLMs for Bio-image Analysis

Summary: 57 use-cases (yet), 17 LLMs (yet), n=10



| Use-case | LLM1 | LLM2 | LLM3 | LLM4 | LLM5 | LLM6 | LLM7 | LLM8 | LLM9 | LLM10 | LLM11 | LLM12 | LLM13 | LLM14 | LLM15 | LLM16 | LLM17 |
|--|------|------|------|------|------|------|------|------|------|-------|-------|-------|-------|-------|-------|-------|-------|
| apply_otsu_threshold_and_count_positive_pixels | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| binary_closing | 1.0 | 0.9 | 0.4 | 1.0 | 0.5 | 0.1 | 0.7 | 0.5 | 0.0 | 0.2 | 0.5 | 0.3 | 0.2 | 0.1 | 0.0 | 0.1 | 0.0 |
| binary_skeleton | 1.0 | 0.9 | 0.8 | 0.9 | 0.9 | 0.2 | 0.1 | 0.3 | 0.2 | 0.0 | 0.2 | 0.0 | 0.2 | 0.0 | 0.1 | 0.0 | 0.1 |
| blend_atlases | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 0.9 | 0.9 | 0.8 | 0.3 | 0.8 | 0.1 |
| combine_columns_of_tables | 1.0 | 0.6 | 0.8 | 0.1 | 1.0 | 0.9 | 1.0 | 0.9 | 0.3 | 0.3 | 0.9 | 0.7 | 0.5 | 0.2 | 0.7 | 0.3 | 0.1 |
| convex_hull_measure_area | 1.0 | 1.0 | 0.9 | 1.0 | 0.7 | 0.8 | 0.0 | 0.2 | 0.9 | 0.6 | 0.4 | 0.3 | 0.2 | 0.2 | 0.2 | 0.0 | 0.2 |
| convolve_image | 1.0 | 0.6 | 0.5 | 0.7 | 0.4 | 0.1 | 0.3 | 0.3 | 0.1 | 0.0 | 0.1 | 0.4 | 0.2 | 0.1 | 0.0 | 0.0 | 0.0 |
| count_number_of_touching_neighbors | 1.0 | 0.6 | 0.6 | 0.1 | 0.2 | 0.1 | 0.1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| count_objects_per_line | 1.0 | 1.0 | 0.5 | 0.3 | 0.3 | 0.1 | 0.4 | 0.4 | 0.0 | 0.0 | 0.4 | 0.4 | 0.3 | 0.5 | 0.2 | 0.2 | 0.1 |
| count_overlapping_regions | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 0.4 | 0.6 | 0.7 | 0.8 | 0.0 | 0.0 | 0.0 | 0.0 | 0.3 | 0.0 | 0.0 | 0.3 |

Summary: 57 use-cases (yet), 17 LLMs (yet), n=10



Benchmarking LLMs for Bio-image Analysis

Common Python libraries (n=570)

Do we need a BiA-specific LLM?

| | | | | | | | | | | | | | | | | | | |
|-------------------------|-------------------|------------------------|------------------------|--------------------|--------------------|--------------------------|--------------------------|----------------------|----------------------------|----------------------------------|---------------------------------|---------------------------------|-----------------------------|------------|--------------------------|-----------|-------------------------|-----|
| numpy | 220 | 442 | 434 | 453 | 398 | 360 | 447 | 460 | 384 | 298 | 478 | 392 | 450 | 426 | 165 | 412 | 454 | 432 |
| scipy | 70 | 118 | 123 | 131 | 141 | 76 | 144 | 156 | 57 | 76 | 168 | 82 | 138 | 118 | 31 | 82 | 114 | 155 |
| skimage | 220 | 102 | 129 | 125 | 132 | 115 | 85 | 98 | 91 | 154 | 118 | 102 | 129 | 151 | 116 | 131 | 96 | 68 |
| cv2 | 0 | 66 | 63 | 44 | 57 | 144 | 107 | 85 | 107 | 43 | 90 | 76 | 107 | 120 | 82 | 31 | 137 | 192 |
| pandas | 60 | 99 | 100 | 99 | 97 | 90 | 100 | 100 | 88 | 74 | 98 | 72 | 99 | 81 | 52 | 89 | 95 | 98 |
| pyclesperanto_prototype | 40 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| vedo | 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| umap | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 16 | 20 | 19 | 20 | 20 |
| dask | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 |
| zarr | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| reference | gpt-4o-2024-05-13 | gpt-4-turbo-2024-04-09 | claude-3-opus-20240229 | gpt-4-1106-preview | gpt-3.5-turbo-1106 | llama3-70b-instruct-q4_0 | llama3-70b-instruct-q8_0 | gemini-1.5-flash-001 | codegemma-7b-instruct-fp16 | mixtral-8x22b-instruct-v0.1-q4_0 | mixtral-8x7b-instruct-v0.1-q5_0 | phi3-3.8b-mini-instruct-4k-fp16 | codellama-70b-instruct-q4_0 | gemini-pro | command-r-plus-104b-q4_0 | codellama | llama3-8b-instruct-fp16 | |

Benchmarking LLMs for Bio-image Analysis

Common error messages (n=570)

Halucinating API?

Forgot import statements?

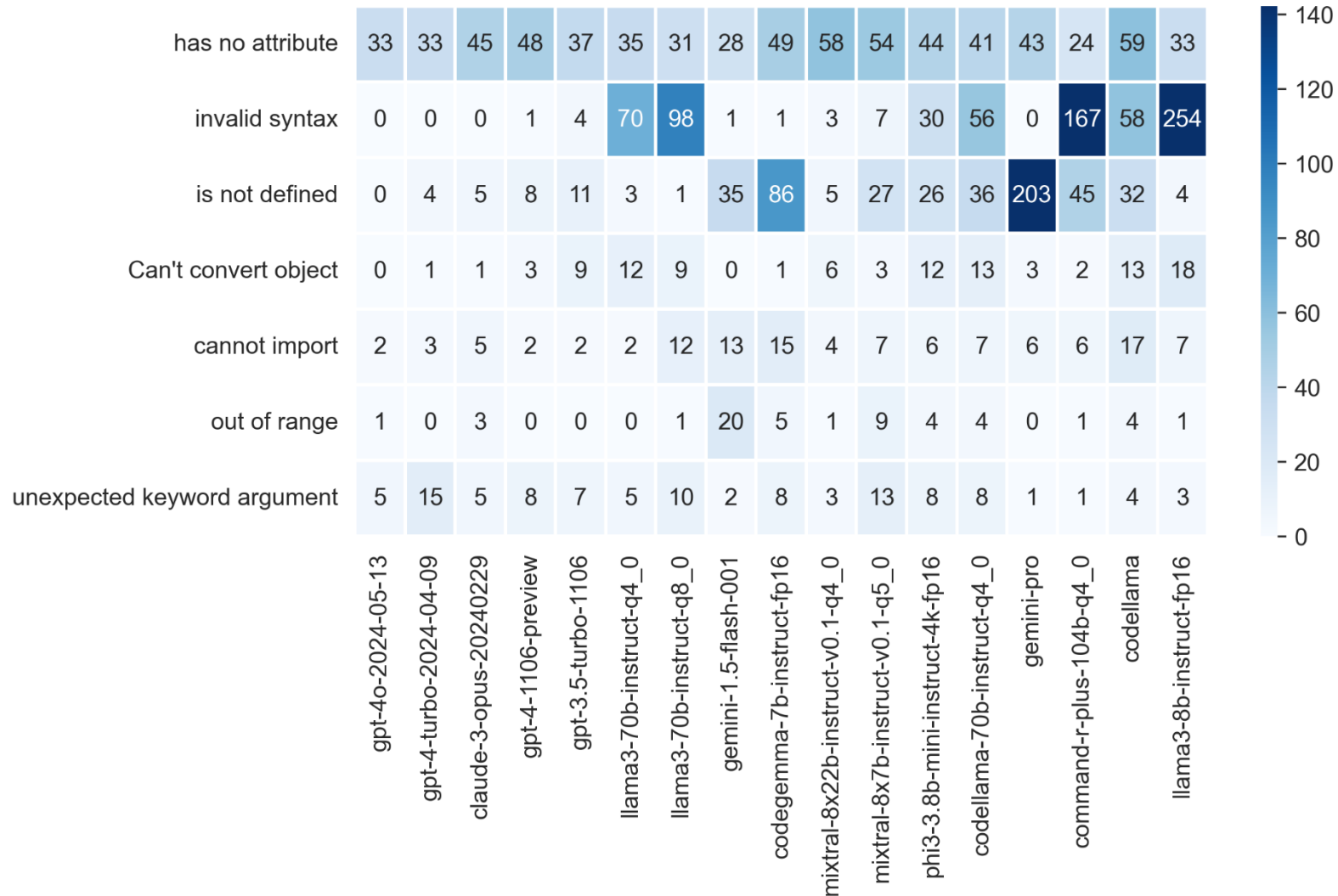
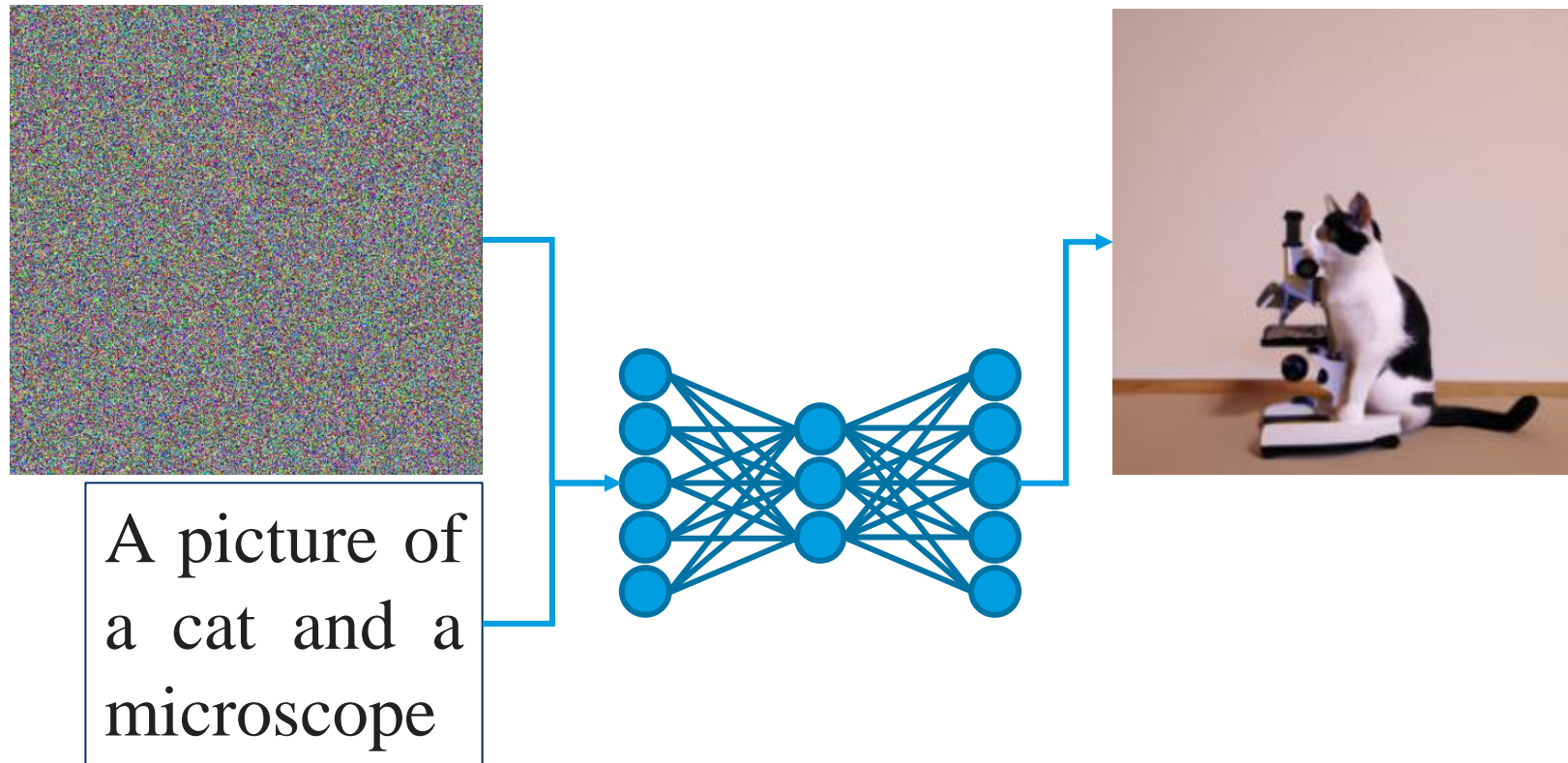


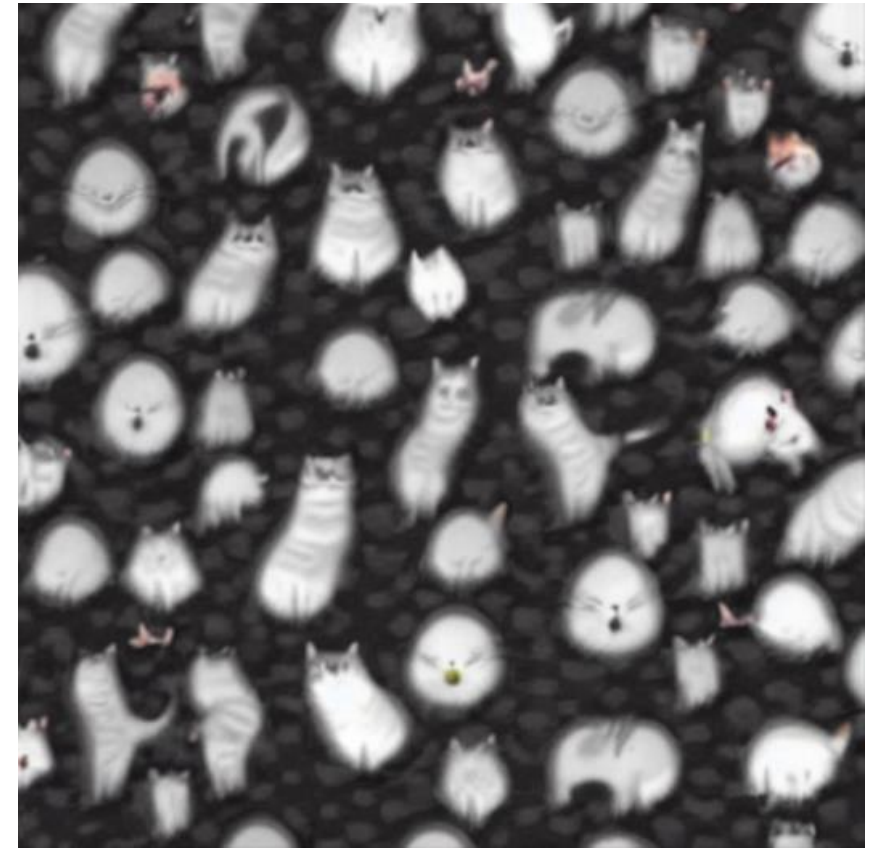
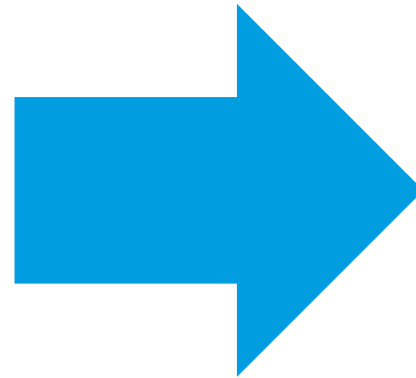
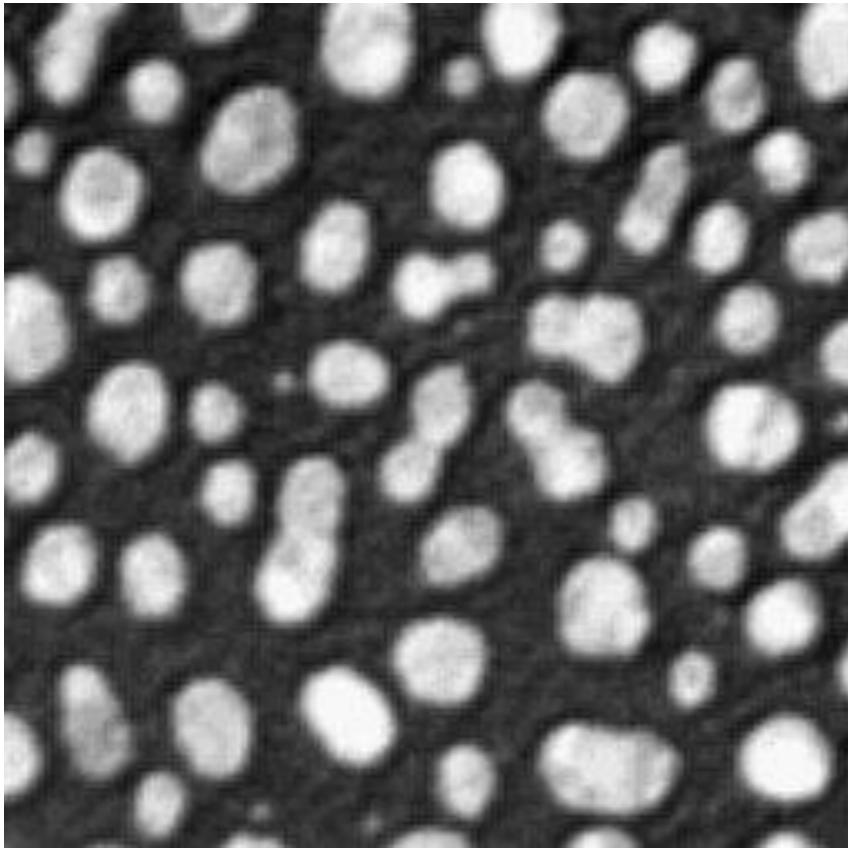
Image generation models

Producing images, also replacing regions in images
(also „Variation“, „Inpainting“, „Gap-filling“, „Replacing“)



Other related use-cases of LLMs

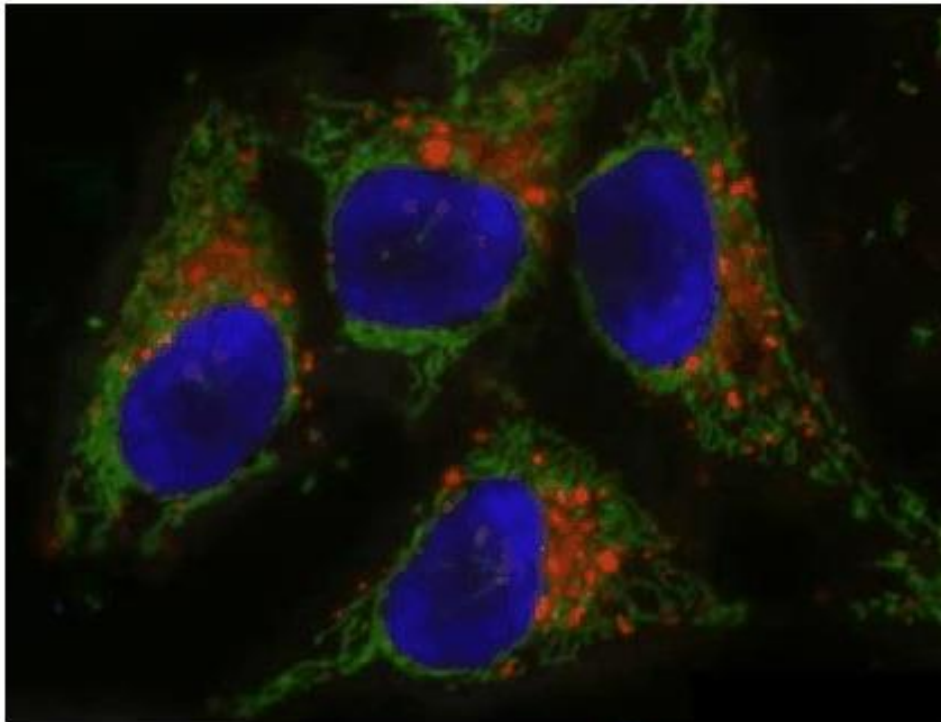
Image variation – just for fun?



Other related use-cases of LLMs

Image inpainting – introducing new challenges

[6]:



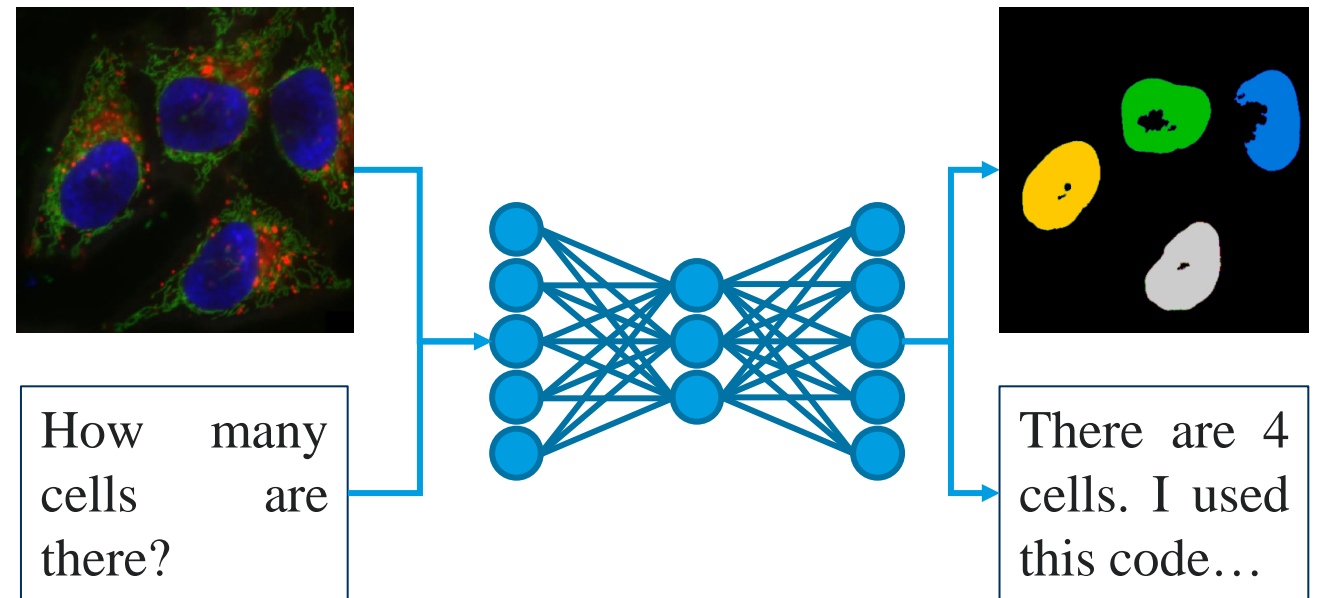
Curtain

672

Detecting
manipulations
becomes harder

Summary & outlook

- LLMs can generate code to analyze biological microscopy images
- Open-source benchmarks can help targeting further improvement
- Challenges:
 - Identify best strategy (fine-tuning, prompt-engineering, RAGs)
 - Multi-modal / multi-agent approaches
 - Introduce good scientific practice -> trustworthy AI



Acknowledgements



Christian Tischer
(EMBL Heidelberg)



Nico Scherf
(MPI CBS Leipzig)



BiAPoL team

- Marcelo Zoccoler
- Johannes Soltwedel
- Maleeha Hassan
- Stefan Hahmann



Former lab members:

- Ryan George Savill
- Laura Zigutyte
- Mara Lampert
- Allyson Ryan
- Conni Wetzker
- Till Korten
- Somashekhar Kulkarni



Jean-Karim Hériché
(EMBL Heidelberg)

Kevin Yamauchi
(ETH Zurich / Basel)

Loic Royer
(CZ Biohub San Francisco)

Matthias Täschner
(ScaDS.AI)



Communities & platforms



NFDI4
BIOIMAGE



image.sc



German
Biolmaging
Gesellschaft für Mikroskopie und Bildanalyse



Funding



Bundesministerium
für Bildung
und Forschung



Diese Maßnahme wird gefördert durch die Bundesregierung aufgrund eines Beschlusses des Deutschen Bundestages. Diese Maßnahme wird mitfinanziert durch Steuermittel auf der Grundlage des von den Abgeordneten des Sächsischen Landtags beschlossenen Haushaltes.

Chan
Zuckerberg
Initiative



Deutsche
Forschungsgemeinschaft



LLMs for Bio-image Analysis
Bio-IAS, Nice, France 2024
Robert Haase @haesleinhuepf
June 14th 2024

RH acknowledges the financial support by the Federal Ministry of Education and Research of Germany and by Sächsische Staatsministerium für Wissenschaft, Kultur und Tourismus in the programme Center of Excellence for AI-research „Center for Scalable Data Analytics and Artificial Intelligence Dresden/Leipzig“, project identification number: ScaDS.AI



TECHNISCHE
UNIVERSITÄT
DRESDEN



UNIVERSITÄT
LEIPZIG